Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 167 (2020) 1739–1748

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Intelligence and Data Science (ICCIDS 2019)

Analyzing Performance of Deep Learning Techniques for Web Navigation Prediction

Honey Jindal[a], Neetu Sardana[b*] , Raghav Mehta[c]

Department of Computer Science and Engineering,
Jaypee Institute of Information Technology, Noida, India

Abstract

The weblog is dynamic and its size is growing exponentially with time in terms of navigation sessions. These stored sessions are used for Web Navigation Prediction (WNP). Each user had varied behavior on the web so is their navigated sessions. With a variety of large dynamic sessions, the task of navigation prediction is becoming challenging. There is a need for an effective method to handle large sessions with multiple labels for predicting user desired information. This paper analyses the performance of Deep Learning techniques like Multi-Layer Perceptron and Long-Short Term Memory based on parameters like number of hidden units, number of layers, activation function, optimization function, learning rate, and batch size. The networks were trained on six experimental parameter setups to form 216 models. The performances of these models are evaluated on two real datasets: BMS and CTI. It has been observed that Long-Short Term Memory performs best on most of the setups.

Keywords: Web; deep; learning; navigation; prediction

*Corresponding Author Email: neetu.sardana@jiit.ac.in

1. Introduction

With an increase in information the weblogs, it is important to segregate relevant and irrelevant information. Web Navigation Prediction [13] [14] is the problem to find an interesting and relevant pattern from user navigated patterns. This problem is large and complex as web navigations are huge and varied. Every user navigation has different labels so problem of handling multiple classes arises. Therefore, multi-class handling models are required

10.1016/j.procs.2020.03.384

to predict user desired information. Deep Learning Neural Networks [15] have the ability to classify large data with multiple classes efficiently.

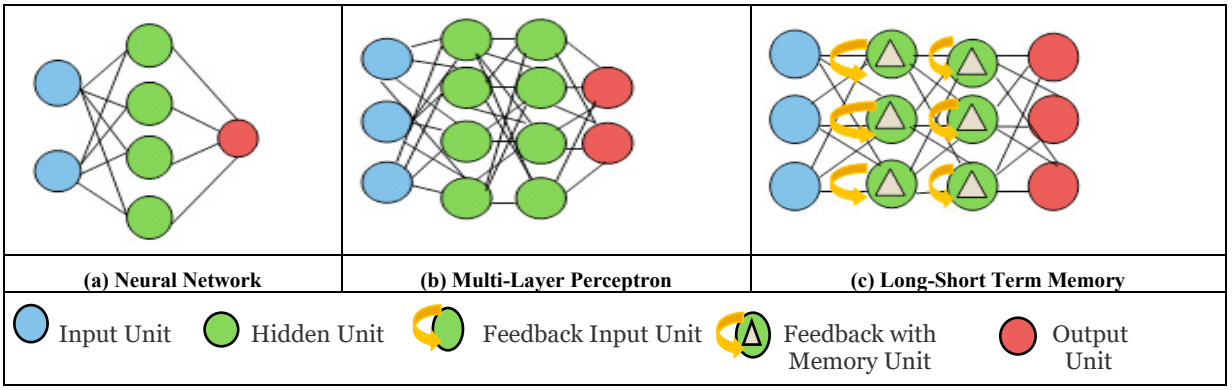

Fig1. Basic architecture of Neural Network and Deep learning Neural Networks

Early neural networks could simulate only a very limited number of neurons at once, so patterns of great complexity could not be recognized. These networks languished through the 1970s. In the mid-1980s, Hinton and others helped spark a revival of interest in neural networks with so-called “deep” models; model with multiple layers of neurons. Thus, the advance neural network has been introduced. To empower the prediction ability of the networks, multiple layers have been added; the resultant model is known as Deep learning model. This paper uses Neural Networks (NNs) [12] and its higher variants because NNs have the ability to learn and model non-linear and complex relations. It is inspired by the human brain. These networks are trained to learn and adapt themselves according to the need. Moreover, they have the ability to generate predictions for unseen data. After learning input data and their relationships, it can generate predictions for unseen data.

Broadly, Deep Neural Networks are classified into two categories: Feed Forward and Feed Backward (or recurrent) neural networks. Feedforward neural network passes the information through successive layers without feedback loops. There are three layers in the basic neural network such as input layer, hidden layer, and output layer. The architecture of the neural network is shown in Fig.1 (a). Feedforward neural networks use a set of neurons to transmit data in the form of input and output values. Feedforward neural networks are used to transfer data by using networks or connections. Adding more hidden layers form a new version of the neural network known as Multi-Layer Perceptron (MLP) [10]. MLP with two hidden layers is shown in Fig. 1(b). Adding multiple layers into a network is known as deep learning. In Feedback or Recurrent neural networks, signals can be transmitted in both directions through network loops. Recurrent networks are powerful and extremely complicated. Computations derived from the input are feedback to the network. This is like a memory present at every neuron. These are dynamic in nature: their states are changing continuously until an equilibrium state is reached. Generally, for complex and large data, Recurrent Neural Network (RNN) is used. RNN learn short term sequences while LSTM has the ability to learn for both long and short term sequences. LSTM [1] architecture is presented in Fig. 1(c).

This paper analyze the performance of Models like Neural Network (NN) , Multiplayer Perceptron (MLP) and Long Short term Memory (LSTM) for Web Navigation Prediction. There are several important parameters which require adjustments for modeling these models like number of hidden units, activation function, optimization function, learning rate, number of epochs and batch size as they all have different range of values. In past studies, NN [17] has been applied for web navigation prediction on a single set of parameter configuration. In this paper varied parameters configurations for NN has been analysed to identify the best combination. In addition, performance of Deep learning models like MLP and LSTM have been analysed for Web Navigation Prediction by varying the parameter values. This paper also compares NN, MLP and LSTM for all varied set of parameters on a common platform to find the best model for predicting user navigation behaviour on web.

1.1. Research Contributions

The main contributions of this paper are given as follows:

1) The paper reports the best parameter configuration for Web Navigation Prediction. The six parameters that have been configured are number of hidden units, activation function, optimization function, learning rate, number of epochs and batch size.
2) The paper conducted exhaustive comparison among Deep Learning networks (MLP and LSTM) and the basic Neural Network by varying the parameter configurations. Six parameter combinations have been chosen to form six different experimental setups. Each model is trained on these setups. These setup forms 6*6*3*2 =216 models where the first six denotes the number of setups, second six denotes N-Grams, three represents the models used and two denotes the datasets.
3) The performance of Deep learning networks for web navigation prediction problem has been analyzed on two real datasets BMS and CTI.
4) The paper reports the best model among Neural Network, Multilayer Perceptron, and Long Short Term Memory.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 describes the parameters and challenges faced for tuning the network. Section 4 presents the experimental results and discussion. Finally, Section 5 concludes the paper.

2. Related Work

The web navigation prediction system classifies user navigated pattern into multiple classes. In past, several machine learning models have been used for this problem. Herein, multi-class classifiers would be suitable as there is large number of classes involved in the web. Deep Learning Neural Networks [15] have the ability to classify large data with multiple classes efficiently. In past work, NN [17] has been used for web navigation prediction on a specific set of parameters. The importance of using those parameter values was not highlighted in the paper. The recent studies on Deep Leaning Models have been used sequence mining in varied applications like speech recognition [18], character recognition [19], handwriting recognition [18] and polyphonic music modelling [18]. Deep learning models were designed to address big problems and it remains unexplored in the web navigation prediction area. Therefore, this paper analyse Deep Learning models like multilayer Perceptron and Long short Term Memory. The paper also investigated the best parameter combinations for these models for Web Navigation Prediction.

3. Neural Network Parameters

In this section, some important parameters and their possible values are used for building the models. The parameters used are number of hidden units, number of layers, activation function, optimization function, learning rate, number of epochs and batch size. The parameters and their possible values are presented in Table 1. This section also explains the strategy that is required to apply while tuning up parameters for Web Navigation Prediction.

- *Number of Hidden Units and layers:* Hidden units are the number of neurons in the hidden layer. The selection of the optimal number of units in the hidden layer is very important because it impacts the model performance. Generally, too few hidden units will leave high training errors due to under-fitting. Too many hidden units will result in low training errors due to over-fitting. It will make the training unnecessarily slow, and will often result in poor generalization. The size of hidden units depends upon the number of input training examples. Hidden layers varies from one to many but for some big applications like distributed systems, cloud computing and image processing more hidden layers are required. Few rules of thumb were defined in [11] for selecting hidden layers.

- *Activation Function:* This is an important parameter of the neural network. Activation function determines the active state of the neuron. It decides whether the information received by the neuron is relevant or not. It transforms the input signal into the non-linear form and sent to the next layer as an input. The activation function is computed as,

$$Y = Activation\ (\sum(weight * X) + bias) \quad (1)$$

where X is the input, weight is the random value of the connection between input neuron to next layer neuron, bias is an extra neuron added to the pre-output layer with value 1.

Table 1: Parameters and their possible values

Parameters	Possible Values
Number of Hidden Units	Varies with application
Number of Layers	Varies with application
Activation Function	Binary Step Function, Linear Function, Sigmoid, Tanh, Relu, Softmax
Optimization Function	Stochastic Gradient Descent, Nesterov accelerated gradient, Adagrad, RMSProp, AdaDelta, Adam
Learning Rate	0.1, 0.001, 0.0001
Number of Epochs	Varies with application
Batch Size	1, 16, 32, 64
Loss	Mean absolute error, Binary cross entropy, Categorial cross entropy

The activation functions used in designing neural networks are Binary Step Function, Linear Function, Sigmoid, Tanh, Relu, Softmax. The binary step function is extremely simple and used for developing a binary classifier. It is not suitable for handling multi-class problems. A linear function is used to activate multiple neurons at the same time. For multi-class, the maximum value can be chosen. The issue of a linear function is that its derivative is constant which means for every back propagation the gradient will remain the same and the network weights will also remains the same. Due to this, it is suitable for simpler problems not for complex problems. Another variant is a sigmoid activation function. It is widely used for smooth and continuously differentiable function. Using this, a small change in the input value X will bring large changes in output Y. The function ranges from 0-1 with S-shaped curve. It works well with back propagation, the error can be propagated and the weights are updated accordingly. Despite their advantages, its value becomes flat at +X and –X regions. This means that gradient value becomes almost zero and the network did not learn. Another issue with this is that its value ranges from 0 to 1 i.e., it is not symmetric around the origin. The values going next neuron are also the same which is not desirable. This issue is addressed by tanh function. It scales the sigmoid function. Tanh function is asymmetric at the origin. Its value ranges from -1 to 1. Similar to the sigmoid function, it has a vanishing gradient problem. Relu is the Rectified Linear unit function. The main advantage of Relu activation function is that it does not activate all the neuron at the same time i.e., a neuron with a negative value will convert into zero or gets deactivated. It makes network sparse and computationally efficient. At the negative side of the graph, its gradient value becomes zero. It implies, during backpropagation, neurons become dead and never gets activated. Softmax is an extension of the sigmoid function used for handling multi-class problems. It maps the output value from 0 to 1. It is ideally used in the output layer of the classifier. There is no thumb rule available for choosing the activation function. The properties of the problem might be able to make a better choice for quick convergence network. Based on the above discussion some properties are listed in [9]. We choose Relu and softmax activation function in our investigations.

- *Optimization:* Optimization is a process which tried to reduce the network error. This plays a crucial role in improving the accuracy of the model. The variants of optimizer are Stochastic Gradient Descent (SGD), Nesterov accelerated gradient, Adagrad, RMSProp, AdaDelta, Adam. SGD is an incremental gradient descent algorithm which tries to find a minimum error via iterations [8]. In each iteration, models generate predictions and compare the predictions with expected outcomes. The difference in predicted value and the real outcome is known as error. This error is used to update network weights and internal model parameters. This updating procedure is followed by the backpropagation algorithm. It does not work well with low learning rate as it

slowed down the learning of the model and high learning rate as it might lead to the oscillations [8]. Moreover, SGD has a hard time escaping the saddle points. To handle saddle points, Adagrad, Adadelta, RMSprop, and ADAM generally preferred. Nesterov accelerated gradient is used to update gradient to the slop and speed up the SGD. AdaGrad performs better than Nesterov accelerated gradient as it performs larger updates for infrequent parameters and smaller updates for the frequent parameters. This, in turn, increases speed, scalability, and robustness of SGD. It is used to train large neural networks. However, the main disadvantage of AdaGrad optimizer is that it makes the learning rate infinitesimally small which degrades model learning ability [8]. To resolve the issue of AdaGrad, two optimizers were developed independently RMSProp and AdaDelta. AdaDelta and RMSProp work similarly but the only difference in Adadelta is that it does not require an initial learning rate constant to start with. Adam is another variant of optimizers which combines good properties of Adadelta and RMSprop. It computes the learning rate for each parameter. Generally, Adam was found to be a good choice as it outperforms RMSProp and Adadelta[7]. Therefore, in this paper, we have chosen Adam and SGD as the optimizers.

- *Learning Rate:* The learning rate is one of the most important parameter which is used to tune the models. It minimizes the error by updating network weights. Choosing too low learning rate and too high learning rate would degrade model performance. The low learning rate will make tiny updates in the network weights and slow down the training process while too high learning rate will cause divergent behavior in the error. In practice, the training should start with high learning rate because in the starting point, random weights are far from optimal and learning rates will fine-grained the network weights by decreasing its value during training. The learning procedure might start with large value i.e., 0.1 and then for the lower values like 0.01, 0.001, etc.
- *Number of epochs:* The number of epochs denotes the number of passes through the training dataset. Each epoch implies that the training sample can update the model internal parameters. It may have one or more batches. The number of epochs can vary from zero to infinity. Generally, a large number of epochs, hundreds or thousands are chosen. This allows the network to reduce the error sufficiently. To choose optimal epoch value, a developer must check the learning curves of error and accuracy. These curves can help to diagnose the learning states of the model i.e., over-learned, under-learned or suitably fit for training.
- *Batch size:* Batch size is a number of training examples passed to the networks. A training dataset can be divided into one or more batches. When all training examples are passed as a single batch then the learning algorithm is known as batch gradient descent [3]. When batch size is one then the learning algorithm is known as stochastic gradient descent [3]. When the batch size is more than one and less than the training size, the learning algorithm is called mini-batch gradient descent [3]. In mini-batch gradient descent, 32, 64 and 128 batch size is more popular [5]. The increase in batch size improves network accuracy [4]. The batch size ranges in between 1 to 1024 with a power of 2. Some important factors to choose mini batch sizes are given in [6]. Nitish Shirish K. et. al [16] has observed that larger batch size degrades the quality of model as it tends to converge to sharp minimizers of the training function. The batch size selection varies with application, after multiple investigations we found two suitable batch size values for WNP as 1 and 64.

In the following section, we will summaries the best parameter value chosen and discuss the experimental results obtained on deep learning neural networks.

4. Results and Discussion

The experiments were carried on GTX 1050 GPU with i5-7400 processor at 3.00 GHz. The system has x64-ased processor with 8.00 GB RAM. The models were trained using Python programming language in Keras environment. The models were evaluated using Tensorflow diagrams. The models were validated using parameters: number of hidden units, number of layers, activation function, optimization function, learning rate, number of epochs and batch size. This section describes the different parameter setups, Real web browsing data sets and experimental results of Neural Network, Deep Learning Networks (Multi layer Perceptron and Long Short Term Memory) on N-grams where N varies from 1 to 6.

4.1 Deep Learning Models and their Parameters

In this section, we present empirical results of neural network(s) and deep learning networks over web navigation prediction datasets. The networks used in the study are Neural Network (NN), Multi-Perceptron Layer (MLP) and Long-Short Term Memory (LSTM). In the neural network, passing whole dataset as an input is not enough [6] to build a successful network. Generally, we need to pass the whole dataset multiple times in the same neural network in order to optimize network weights. In addition, the performance of the neural network may vary with the dataset used. The dataset may have under-fitted, over-fitted or will have optimal learning curve [2]. Therefore, the correct choice of network parameters is required for developing a model. This study analyses model performance using different combinations of parameters- loss, optimizer, activation function, learning rates, batch size, number of epochs.It has been found in the past that through learning rates 0.0005, 0.001, 0.00146 networks converges faster than other learning rates [6]. Using the learning rate above 0.001 [6] increases the training time and also increases the variance of the training time. Therefore, we initiate investigations with the learning rate of 0.001. To conduct experiments we use categorical_crossentropy loss, Adam and SGD optimizers, Relu and softmax activation functions, 1 and 64 batch size. We conduct an experiment on 20 epochs to analyze the nature of neural networks.

Table 2 Neural Networks Setup

Setup	Loss Function	Optimizer	Activation Function	Learning rate	Drop Out	Batch Size	Output Layer
1	Categorical Cross entropy	Adam	Relu	0.001	True	64	Softmax
2	Categorical Cross entropy	Adam	Relu	0.001	True	1	Softmax
3	Categorical Cross entropy	SGD	Relu	0.001	True	64	Softmax
4	Categorical Cross entropy	SGD	Relu	0.001	True	1	Softmax
5	Categorical Cross entropy	SGD	Softmax	0.001	True	64	Softmax
6	Categorical Cross entropy	SGD	Softmax	0.001	True	1	Softmax

Six parameter combination setups have been used in the paper which is shown in Table 2. The experiments were carried over two web navigation datasets: BMS and CTI. These datasets have user navigation sessions of varied length. For simplicity, N-Gram sliding window concept has been used to generate the training and testing sessions. There are 216 models formed from these setups. Sections describe the performance of models formed.

4.2 Datasets

The performance of deep learning neural networks is evaluated on two real datasets, BMS and CTI.

- *BMS WebView1:* This dataset is recorded from an e-commerce website, Gazelle.com. The dataset was used in the KDD Cup 2000 competition. It contains 59,601 sessions and contains 497 distinct items or web pages.
- *CTI Dataset:* This dataset contains pre-processed navigations derived from the CTI University website. The data are collected during the two weeks of April 2002. The processed data contains 13745 sessions and 683 web pages. The dataset consists of 16-page categories and each category is labeled as numeric value i.e., 1 for search, 2 for the program and so on.

The details of the datasets used are summarized in Table 3.

Table 3: Dataset Summary

Datasets	Source	Year	Sessions	Web Pages
BMS WebView1	www.gazelle.com	2000	59601	497
CTI	www.cs.depaul.edu	2002	13745	683

4.3 Results

In this section, we present the experimental results of Deep Learning neural networks over varied parameters combination. Overall six setups and 216 models have been formed using these combinations.

Table 4(A-F) presents prediction accuracy and validation loss of each model varying with N-Grams. It has been observed from Table 4(A-E) that LSTM is performing superior to Feed forward models, MLP and NN on both datasets. The improvement has been noticed in prediction accuracy with minimal loss. For Setup 3 and 4, the results are shown in Table 4 (C-D), it can be observed clearly that the prediction accuracy of all the models is very less. The prediction accuracy of BMS dataset is below 10 in all variants of neural networks. In Setup 4, LSTM has higher prediction accuracy than Setup 3 LSTM. Setup 3 and 4 are found to be inappropriate parameters for web navigation prediction. Table 4 (E) depicts that there is a slight difference observed in the performance of lower order neural networks when applied on BMS dataset. LSTM is performing better for very short (1-gram) and long sessions (4- 6 grams). This proved the property of LSTM which can learn for short and long patterns both. In CTI dataset, LSTM is performing better on all N-Grams. In Setup 6, MLP is performing better than LSTM and NN on both datasets.

Although LSTM performance is better in almost all the setups considered but it has attained best prediction accuracy with minimal loss in setup1 with parameters categorical cross entropy, Adam Optimizer, Relu activation function, 0.001 learning rate, Batch size 64 and Softmax output activation function. The improvement range of LSTM over Neural Network in Setup1 for BMS dataset is 6.74 to 19.21. LSTM improvement over MLP for BMS dataset is 1.571 to 20.331 in Setup1. LSTM improvement over basic Neural Network in Setup1 for CTI dataset ranges from 10.48 to 38.45 and over MLP ranges from 11.92 to 52.75. In feedforward networks, MLP is performing better at setup1 with parameters categorical cross entropy, SGD Optimizer, Softmax activation function, 0.001 learning rate, Batch size 1, Drop out true and Softmax output activation function. The improvement of MLP over NN ranges from 2.04 to 7.71 on BMS dataset and 0.88 to 5.26 on CTI Dataset.

Table 4: Prediction Accuracy of Neural Network and Deep Learning Models

(A) Setup 1

(loss: Categorical cross entropy, Optimizer: Adam, Activation Function: Relu, Learning rate:0.001, Batch size:64, Drop Out:true, Output Activation Function: Softmax)

	BMS Dataset			CTI Dataset		
Grams	NN	MLP	LSTM	NN	MLP	LSTM
1	[40.15, 5.3]	[24.29, 6.882]	[39.47, 5.027]	[14.35, 7.765]	[0.0579, 14.79]	[52.80, 2.913]
2	[20.56, 8.112]	[10.30, 12.38]	[30.63, 5.608]	[14.52, 8.599]	[17.42, 8.322]	[31.06, 4.205]

3	[6.283, 13.38]	[13.02, 12.95]	[25.49, 6.501]	[7.888, 14.96]	[3.743, 14.66]	[25.72, 5.070]
4	[8.575, 15.23]	[20.80, 9.397]	[22.37, 7.124]	[2.999, 14.99]	[2.121, 15.73]	[25.77, 5.945]
5	[4.889, 15.34]	[1.956, 15.70]	[15.64, 7.699]	[9.738, 14.66]	[0.0499, 16.05]	[20.21, 6.981]
6	[4.189, 16.07]	[3.514, 15.75]	[10.92, 7.923]	[2.500, 15.99]	[2.500, 15.90]	[14.42, 7.770]

(B) Setup 2

(loss:Categorical cross entropy, Optimizer:Adam, Activation Function:Relu, Learning rate:0.001, Batch size:1, Drop Out:true, Output Activation Function:Softmax)

	BMS Dataset			CTI Dataset		
Grams	NN	MLP	LSTM	NN	MLP	LSTM
1	[4.517, 7.502]	[37.23, 16.12]	[14.10, 5.343]	[29.72, 5.019]	[5.658, 16.11]	[29.41, 5.040]
2	[7.66, 7.311]	[2.842, 16.11]	[30.82, 7.304]	[10.97, 6.231]	[8.725, 16.10]	[18.23, 6.243]
3	[7.671, 7.48]	[12.98, 16.11]	[31.66, 8.052]	[9.759, 6.271]	[1.159, 16.11]	[18.23, 6.602]
4	[8.227, 7.931]	[0.0579, 16.10]	[30.53, 7.986]	[12.66, 6.139]	[0.0585, 15.97]	[26.34, 6.655]
5	[9.511, 8.262]	[20.71, 15.99]	[25.07, 9.297]	[14.61, 6.461]	[6.742, 15.39]	[25.47, 7.270]
6	[9.595, 8.493]	[2.297, 15.67]	[26.35, 9.29]	[14.79, 6.437]	[4.167, 15.51]	[20.00, 8.649]

(C) Setup 3

(loss:Categorical cross entropy, Optimizer:SGD, Activation Function:Relu, Learning rate:0.001, Batch size:64, Drop Out:true, Output Activation Function:Softmax)

	BMS Dataset			CTI Dataset		
Grams	NN	MLP	LSTM	NN	MLP	LSTM
1	[4.372, 7.010]	[3.964, 7.712]	[8.802, 6.132]	[5.351, 5.687]	[5.351, 5.554]	[40.29, 5.955]
2	[5.129, 8.092]	[3.597, 7.984]	[7.66, 5.937]	[1.875, 7.269]	[2.931, 7.749]	[11.11, 6.024]
3	[2.395, 7.757]	[0.079, 7.928]	[8.4, 5.859]	[0.053, 6.942]	[0.0757, 7.144]	[9.715, 6.191]
4	[2.491, 8.141]	[2.260, 8.557]	[9.212, 6.007]	[1.097, 7.528]	[2.195, 7.406]	[13.46, 6.138]
5	[1.867, 8.259]	[6.933, 8.226]	[10.67, 6.113]	[0.074, 7.276]	[0.0873, 8.072]	[16.23, 6.104]
6	[0.081, 8.380]	[1.081, 8.428]	[9.865, 6.103]	[2.083, 7.612]	[0.0416, 8.251]	[10.61, 6.337]

(D) Setup 4

(loss:Categorical cross entropy, Optimizer:SGD, Activation Function:Relu, Learning rate:0.001, Batch size:1, Drop Out:true, Output Activation Function:Softmax)

	BMS Dataset			CTI Dataset		
Grams	NN	MLP	LSTM	NN	MLP	LSTM
1	[4.886, 8.119]	[4.004, 7.172]	[6.967, 6.280]	[25.09, 6.100]	[18.85, 5.352]	[31.53, 4.802]
2	[3.752, 7.530]	[3.131, 7.481]	[26.44, 5.459]	[0.0954, 7.52]	[3.033, 6.539]	[27.57, 5.076]
3	[1.562, 8.432]	[3.436, 7.784]	[29.36, 5.897]	[0.0267, 6.416]	[0.0267, 7.026]	[27.32, 5.071]
4	[2.202, 8.223]	[5.388, 8.411]	[23.70, 6.171]	[0.0292, 7.429]	[1.024, 7.474]	[28.53, 5.237]
5	[1.156, 10.33]	[3.644, 8.673]	[22.58, 6.206]	[0.0499, 7.112]	[0.074, 6.972]	[26.09, 5.331]
6	[4.459, 9.197]	[2.973, 9.311]	[19.86, 6.334]	[0.0416, 8.075]	[2.083, 7.616]	[18.75, 5.692]

(E) Setup 5

(loss:Categorical cross entropy, Optimizer:SGD, Activation Function:Softmax, Learning rate:0.001, Batch size:64, Drop Out:true, Output Activation Function:Softmax)

	BMS Dataset			CTI Dataset		
Grams	NN	MLP	LSTM	NN	MLP	LSTM
1	[7.362, 6.118]	[7.154, 6.047]	[7.691, 6.147]	[29.38, 6.006]	[29.38, 5.437]	[40.18, 5.974]
2	[7.726, 6.129]	[7.371, 6.072]	[7.66, 5.928]	[10.74, 6.426]	[10.97, 6.293]	[10.97, 6.039]
3	[11.21, 6.156]	[6.560, 6.148]	[11, 5.893]	[7.843, 6.493]	[8.690, 6.492]	[10.70, 6.182]
4	[6.663, 6.187]	[7.068, 6.170]	[9.676, 6.063]	[9.290, 6.501]	[9.802, 6.501]	[13.68, 6.133]
5	[5.193, 6.194]	[7.289, 6.185]	[10.49, 6.057]	[2.622, 6.514]	[6.742, 6.510]	[14.48, 6.136]
6	[1.891, 6.201]	[2.297, 6.196]	[9.784, 5.992]	[1.042, 6.524]	[0.0833, 6.521]	[11.67, 6.386]

(F) Setup 6

(loss:Categorical cross entropy, Optimizer:SGD, Activation Function:Softmax, Learning rate:0.001, Batch size:1, Drop Out:true, Output Activation Function:Softmax)

	BMS Dataset			CTI Dataset		
Grams	NN	MLP	LSTM	NN	MLP	LSTM
1	[32.27, 5.218]	[39.56, 4.632]	[7.22, 6.284]	[56.12, 3.305]	[58.55, 3.107]	[31.56, 4.783]
2	[29.91, 5.437]	[36.72, 4.847]	[28.24, 5.562]	[31.18, 4.555]	[35.48, 4.180]	[29.93, 4.897]

3	[23.74, 5.664]	[31.45, 5.352]	[30.48, 5.816]	[24.37, 4.992]	[29.01, 4.606]	[24.73, 5.042]
4	[16.98, 6.042]	[23.81, 5.754]	[23.80, 6.114]	[23.27, 5.310]	[28.53, 4.912]	[27.94, 5.149]
5	[14.22, 6.159]	[17.87, 5.995]	[17.82, 6.33]	[19.10, 5.487]	[21.10, 5.319]	[21.01, 5.401]
6	[10.68, 6.280]	[12.72, 6.219]	[12.70, 6.369]	[15.37, 5.841]	[16.25, 5.809]	[0.88, 5.690]

5. Conclusion

Web Navigation Prediction is a complex problem which requires multiple class handling. Deep Learning Neural Networks have the ability to handle large data with multiple classes efficiently. This paper investigated the performance of the basic neural network with deep learning feed forward network (Multi-Layer Perceptron) and deep learning feed backward network (Long-Short Term Memory) for Web Navigation Prediction. These networks are evaluated using different combinations of parameters in six setups and 216 models were evaluated. The experiments were carried on two real web navigation datasets, BMS and CTI. It has been observed that the performance of LSTM is superior in most of the setups. The parameters considered in setup 1 were the best as LSTM attained best prediction accuracy with minimal loss. While comparing MLP with NN, the performance was best in setup 6 and MLP has shown better results. In future, the deep learning models can be hybridized with web navigation prediction sequential models to address unseen predictions.

References

[1] Hochreiter, S. & Schmidhuber, J. (1997) " Long Short-Term Memory", *Neural Computation* **9(8):** 1735–1780.
[2] Sagar Sharma, "Epoch vs Batch Size vs Iterations', https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9, (accessed on 8th January 2019).
[3] Jason Brownlee, " What is a difference between batch and epoch in a neural network? ", https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch, (accessed on 8th January 2019).
[4] Pavlo M. Radiuk (2017) "Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets" *Information Technology and Management Science* **20:** 20–24.
[5] Kien Huynh, "What is the usual batch sizes people used to train neural nets?", https://www.quora.com/What-are-the-usual-batch-sizes-people-use-to-train-neural-nets, (accessed on 9th January 2019).
[6] David Mack, "How to pick the best learning rate for your machine learning project", https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2, (accessed on Januray 2019).
[7] Sebastian Ruder,"An overview of gradient descent optimization algorithms", http://sebastianruder.com/optimizing-gradient-descent/index.html, (accessed on January 2019).
[8] Bikal Basnet, "LSTM optimizer choice?", https://deepdatascience.wordpress.com/2016/11/18/which-lstm-optimizer-to-use/, (accessed on January 2019).
[9] Disha Shree Gupta, "Fundamenetals of Deep Learning-Activation Functions and when to use them?",https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/(accessed on February 2019).
[10] DeapLearning 0.1 documentation, "http://deeplearning.net/tutorial/mlp.html", Theano Development team, 2008-2013.
[11] Jeff Heaton, "The number of hidden layers", https://www.heatonresearch.com/2017/06/01/hidden-layers.html, (accessed on February 2019).
[12] Jose C. Principe et. al, "Neural and Adaptive Systems: Fundamentals Through Simulation", John Wiley & Sons, 1999.
[13] Jindal H., Sardana N. (2018) "Decomposition and Compression of Backward Browsing for Efficient Session Regeneration and Navigation Prediction", *International Journal of Web Based Communities* **14(2):** 196-223.
[14] Jindal H., Sardana N.(2017) "Empirical analysis of Web Navigation Prediction Techniques", Journal of Cases on Information Technology**19(1):**1-14.
[15] Ian Goodfellow and Yoshua Bengio and Aaron Courville, "Deep Learning", MIT Press, 2016.
[16] Nitish Shirish K. et. al (2017) "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima.", ICLR Conference.
[17] Mamoun A. Awad and Latifur R. Khan (2007) "Web navigation prediction using multiple evidence combination and domain knowledge" *IEEE Transaction on systems, man and cybernetics- Part A: Systems and Humans* **37(6):** 1054-1142.
[18] Klaus Greff, R. K. et. al (2017) "LSTM: A search space odyssey" *Transaction on Neural Networks and Learning Systems*. arXiv: 1503.04069v2.
[19] Amit Ch. Et al. (2010) "Influence of introducing an additional hidden layer on the character recognition capability of a BP neural network having one hidden layer" *International Journal of Engineering and Technology***2(1):** 24-28.